

Minimizing Total Calibration Cost

Eric Angel* Evripidis Bampis† Vincent Chau‡ Vassilis Zissimopoulos§

Abstract

Bender et al. (SPAA 2013) have proposed a theoretical framework for testing in contexts where safety mistakes must be avoided. Testing in such a context is made by machines that need to be often calibrated. Given that calibration costs, it is important to study policies minimizing the calibration cost while performing all the necessary tests. We focus on the single-machine setting and we extend the model proposed by Bender et al. by considering that the jobs have arbitrary processing times and that the preemption of jobs is allowed. For this case, we propose an optimal polynomial time algorithm. Then, we study the case where there are several types of calibrations with different lengths and costs. We first prove that the problem becomes NP-hard for arbitrary processing times even when the preemption of the jobs is allowed. Finally, we focus on the case of unit-time jobs and we show that a more general problem, where the recalibration of the machine is not instantaneous but takes time, can be solved in polynomial time.

1 Introduction

The scheduling problem of minimizing the number of calibrations has been recently introduced by Bender et al. in [4]. It is motivated by the Integrated Stockpile Evaluation (ISE) program [1] at Sandia National Laboratories for testing in contexts where safety mistakes may have serious consequences. Formally, the problem can be stated as follows: we are given a set \mathcal{J} of n jobs (tests), where each job j is characterized by its release date r_j , its deadline d_j and its processing time p_j . We are also given a (resp. a set of) testing machine(s) that must be calibrated in a regular basis. In the simplified model of Bender et al. the calibration of a machine has a unit cost, and it is instantaneous, i.e., a machine can be recalibrated between the execution of two unit-time jobs that are processed in consecutive time-units. A machine stays calibrated for T time-units and a job can only be processed during an interval where the machine is calibrated. The goal is to find a feasible schedule performing all the tests (jobs) between their release dates and deadlines and minimizing the number of calibrations. Using the classical three-field notation in scheduling [6], the problem can be denoted as $P | r_j, d_j, T | (\# \text{ calibrations})$. Bender et al. studied both the single-machine and multiple-machine problems. For the single-machine case, they showed that there is a polynomial-time algorithm, called the Lazy Binning algorithm that solves the problem optimally for unit-time jobs. For the multiple-machine case with unit-time jobs, they proposed a 2-approximation algorithm. However, the complexity status of the multiple-machine case with unit-time jobs remains open. Bender et al. in their paper [4] stated that “*it would be interesting to generalize the model in the case where the jobs may have not unit size, or when calibrations may not be instantaneous, but may require machine time*”.

Fineman and Sheridan [5] studied a first generalization of the problem by considering that the jobs have arbitrary processing times. They focused on the multiple-machine-non-preemptive case where the execution of a job is not allowed to be interrupted once it has been started. Given

*IBISC ; Université d'Évry Val d'Essonne, Évry, France

†Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, Paris, France

‡Department of Computer Science, City University of Hong Kong

§Department of Informatics & Telecommunications, National and Kapodistrian University of Athens

that the feasibility problem is NP-hard, they considered a *resource-augmentation* [7] version of the problem. They were able to relate this version with the classical *machine-minimization* problem [10] in the following way: suppose there is an s -speed α -approximation algorithm for the machine-minimization problem, then there is an $O(\alpha)$ -machine s -speed $O(\alpha)$ -approximation for the resource-augmentation version of the problem of minimizing the number of calibrations. Here we consider some additional generalizations of the model of Bender et al. More precisely, we consider that: (i) the jobs may have arbitrary processing times and that the preemption of the jobs is allowed, and/or (ii) there are several types of calibrations, and/or (iii) the calibrations are not instantaneous but require machine time. For these generalizations, we focus on the complexity status of the associated optimization problem and we propose either an optimal polynomial time algorithm or an NP-hardness proof.

Articulation of the paper and our contribution. In Section 2, we consider the case where the jobs have arbitrary processing times and the preemption, i.e. the possibility to interrupt the execution of a job and resume it later, is allowed. We denote this problem as $1 \mid r_j, d_j, pmtn, T \mid (\# \text{ calibrations})$. We present an optimal quadratic-time algorithm for the problem. Then, in Section 3, we study the case of scheduling a set of jobs when K different types of calibrations are available. Each calibration type is associated with a length ℓ_i and a cost f_i . The objective is to find a feasible schedule minimizing the total calibration cost. We show that the problem, denoted as $1 \mid r_j, d_j, pmtn, \{\ell_1, \dots, \ell_K\} \mid cost(calibrations)$ is NP-hard.

Then, we enrich the model by considering that the calibrations are not instantaneous. More precisely, we consider that every calibration takes λ units of time during which the machine cannot be used. Moreover, we allow the recalibration of the machine at any time (even if the machine was already calibrated at this moment). This assumption is necessary in order to avoid infeasibility. To see this consider the following instance (Figure 1). The machine has to be calibrated at time 0 and requires $\lambda = 3$ units of time for being available for the execution of jobs. At time 3 the machine is ready to execute job 1 and it remains calibrated for $T = 4$ time units. If we have not the possibility to recalibrate an already calibrated machine then the earliest time at which we can calibrate the machine is at time 7. This would lead to the impossibility of executing job 2. However, a recalibration at time 4 would lead to a feasible schedule.

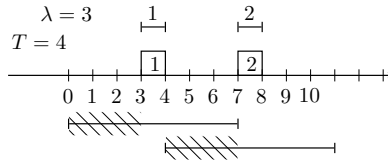


Figure 1: An infeasible instance if we do not allow to recalibrate at any time. There is a single machine and a single type of calibration of length $T = 4$. The activation time, i.e. the time that is required in order to the calibration to be effective is $\lambda = 3$.

Given that the problem with arbitrary processing times is NP-hard in the presence of many (instantaneous) calibration types even when the preemption of jobs is allowed, we focus on the case of unit-time jobs. In Section 4, we prove that the problem with unit-time jobs, denoted as $1 \mid r_j, d_j, p_j = 1, \lambda + \{\ell_1, \dots, \ell_K\} \mid cost(calibrations)$, can be solved in polynomial time using dynamic programming.

A related problem. A scheduling problem which is somehow related to the problem of minimizing the calibration cost is the problem where the goal is the minimization of the number of gaps (idle periods) [2], [3]. The difference is that in the gap-minimization version, the cost of starting a constant-sized period of activity is not taken into account.

2 Arbitrary processing times and preemption

We suppose here that the jobs have arbitrary processing times and that the preemption of the jobs is allowed. An obvious approach in order to obtain an optimal preemptive schedule is to divide each job j into p_j unit-time jobs with the same release date and deadline as job j and then apply the Lazy Binning (LB) algorithm of [4] that optimally solves the problem for instances with unit-time jobs. However, this idea leads to a pseudopolynomial-time algorithm. Here, we propose a more efficient way for solving the problem which is also based on the idea of Lazy Binning. Before introducing our algorithm, let us first recall the idea of Lazy Binning: at each iteration we fix a date t and we schedule the (remaining) jobs starting at time $t + 1$ using the Earliest Deadline First (EDF) policy¹. If a feasible schedule exists (for the remaining jobs), we update t to $t + 1$, otherwise we set the next calibration to start at time t which is called the current *latest-starting-time* of the calibration. Then, we remove the jobs that are scheduled during this calibration interval and we iterate after updating t to $t + T$, where T is the calibration length. The polynomiality of the algorithm for unit-time jobs comes from the observation that the starting time of any calibration is at a distance of no more than n time-units before any deadline. In our case however, i.e. when the jobs have arbitrary processing times, a calibration may start at a distance of at most $P = \sum_{j=1}^n p_j$ time-units before any deadline.

Definition 1. Let $\Psi := \bigcup_i \{d_i - P, \dots, d_i\}$ where $P = \sum_{j=1}^n p_j$.

Proposition 1. *There exists an optimal solution in which each calibration starts at a time in Ψ .*

Proof. Without loss of generality, we assume that jobs have unit processing times, otherwise we could replace each job j by p_j unit-time jobs. Let σ be an optimal solution in which there is at least one calibration that does not start at a time in Ψ . We show how to transform the schedule σ into another optimal schedule that satisfies the statement of the proposition.

Let c_j be the first calibration of σ that starts at time $t' \notin \Psi$. Let c_j, \dots, c_i be the maximum set of consecutive calibrations such that when a calibration finishes another starts immediately. We denote by c_{i+1} the next calibration that is not adjacent to calibration c_i . We can push the

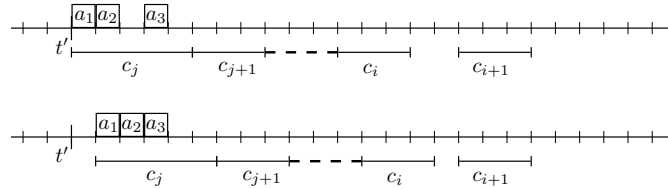


Figure 2: Illustration of Proposition 1. The first schedule is an optimal schedule. The second one is obtained after pushing the continuous block of calibrations c_j, \dots, c_i to the right.

set of calibrations c_j, \dots, c_i into the right (we delay the calibrations) until:

- either we reach the next calibration c_{i+1} ,
- or c_j starts at a time in Ψ .

Note that this transformation is always possible. Indeed, since c_j starts at a time that is in a distance more than P from a deadline, it is always possible to push the scheduled jobs to the right. In particular, if there is no jobs scheduled when calibration c_j starts, then there is no modifications for the execution of jobs. Otherwise, there is at least one job scheduled when calibration c_j starts. Let us denote a_1, \dots, a_e be the continuous block of jobs. Since the starting time of job a_1 is at a distance more than P from a deadline (to the left hand side), then all

¹an EDF schedule is a schedule in which at any time, the job with the smallest deadline among the available jobs is scheduled first.

Algorithm 1 Preemptive Lazy Binning

```
1: Jobs in  $\mathcal{J}$  are sorted in non-decreasing order of deadline
2: while  $\mathcal{J} \neq \emptyset$  do
3:    $t \leftarrow \max_{i \in \mathcal{J}} d_i$ ,  $k \leftarrow 0$ 
4:   for  $i \in \mathcal{J}$  do
5:     if  $t > d_i - \sum_{j \leq i, j \in \mathcal{J}} p_j$  then
6:        $t \leftarrow d_i - \sum_{j \leq i, j \in \mathcal{J}} p_j$ 
7:        $k \leftarrow i$ 
8:     end if
9:   end for
10:  //Schedule jobs  $\{j \leq k \mid j \in \mathcal{J}\}$  from  $t$  to  $d_k$  by applying the EDF (Earliest Deadline
    First) policy and remove them from  $\mathcal{J}$ .
11:   $u \leftarrow t + \left\lceil \frac{d_k - t}{T} \right\rceil \times T$ 
12:  Calibrate the machine at time  $t, t + T, t + 2T, \dots, u - T$ 
13:  Schedule fragment of jobs from  $k + 1, \dots, n$  in  $[d_k, u)$  in EDF order
14:  Let  $q_j$  for  $j = k + 1, \dots, n$  the processed quantity in  $[d_k, u)$ 
15:  //Update processing time of jobs in the following way
16:  for  $i = k + 1, \dots, n$  do
17:     $p_i \leftarrow p_i - q_i$ 
18:    if  $p_i = 0$  then
19:       $\mathcal{J} \leftarrow \mathcal{J} \setminus i$ 
20:    end if
21:  end for
22: end while
```

these jobs can be pushed to the right by one unit. This transformation is possible given that no job of this block finishes at its deadline. Note that after this modification, jobs can be assigned to another calibration.

We can repeat the above transformation until we get a schedule satisfying the statement of the proposition. \square

For jobs with arbitrary processing times when the preemption of the jobs is allowed, we propose the following algorithm whose idea is based on the Lazy Bining algorithm: we first compute the current latest-starting-time of the calibration such that no job misses its deadline (this avoids to consider every date in Ψ). This calibration date depends on some deadline d_k . At each iteration, among the remaining jobs, we compute for every deadline the sum of the processing times of all these jobs (or of their remaining parts) having a smaller than or equal deadline and we subtract it from the current deadline. The current latest-starting-time of the calibration is obtained by choosing the smallest computed value. Once the calibration starting time is set, we schedule the remaining jobs in the EDF order until reaching d_k and we continue to schedule the available jobs until the calibration interval finishes. In the next step, we update the processing time of the jobs that have been processed. We repeat this computation until there is no processing time left. A formal description of the algorithm, that we call the Preemptive Lazy Binning (PLB) algorithm, is given in the frame Algorithm 1.

We can prove the optimality of this algorithm using a similar analysis as the one for the Lazy Binning algorithm in [4].

Proposition 2. *The schedule returned by Algorithm PLB is a feasible schedule in which the starting time of each calibration is maximum.*

Proof. The condition in line 5 in Algorithm PLB ensures that we always obtain a feasible schedule. In fact, we compute the latest-starting-time at each step and this date is exactly the

latest date of the first calibration.

By fixing a deadline d_i , we know that jobs that have a deadline lower than d_i have to be scheduled before d_i , while the other jobs are scheduled after d_i . When we update t for every deadline d_i in the algorithm, we assume that there is no idle time between $d_i - \sum_{j \leq i, j \in \mathcal{J}} p_j$ and d_i . For the sake of contradiction, suppose that a feasible schedule in which there exists a calibration that is not started at a date computed by the algorithm. We show that the starting time of this calibration is not maximum, we denote this date t . Since, the starting time of the calibration is not one of $d_i - \sum_{j \leq i, j \in \mathcal{J}} p_j \forall i$, then there is at least one unit of idle time between the starting time of the calibration and some deadline d_i . Finally, we can delay all calibrations starting at t or after as well as the execution of the jobs inside these calibrations by keeping the EDF order. This can be done in a similar way as in the proof of Proposition 1. \square

Proposition 3. *Algorithm PLB is optimal.*

Proof. It is sufficient to prove that Algorithm PLB returns the same schedule as Lazy Binning after splitting all jobs to unit-time jobs. We denote respectively ALG and LB the two schedules returned by the algorithms.

Let t the first time at which the two schedules differ. The jobs executed before t are the same in both schedules since the jobs are scheduled in the EDF order. Since the schedules are the same before t , the remaining jobs are the same after t . Two cases may occur:

- a job is scheduled in $[t, t+1)$ in ALG and not in LB . This means that the machine is not calibrated at this time slot in the schedule produced by LB . Since the calibrations are the same before t in both schedules, then a calibration starting at t is necessary in ALG . Thanks to Proposition 2, we have a contradiction to the fact that we were looking for the latest-starting-time of the calibration.
- a job is scheduled in $[t, t+1)$ in LB and not in ALG . This means that there does not exist a feasible schedule starting at $t+1$ with the remaining jobs. Hence, ALG is not feasible. This case cannot happen thanks to Proposition 2. \square

Proposition 4. *Algorithm PLB has a time complexity in $O(n^2)$.*

Proof. We first sort jobs in non-decreasing order of deadline in $O(n \log n)$ time. At each step, we compute the first latest-time of the calibration in $O(n)$ time. Then the scheduling of jobs in the EDF order takes $O(n)$ time. Then, we need to update the processing times of the jobs whose execution has been started. This can be done in $O(n)$ time. At each step, we schedule at least one job. Hence, there are at most n steps. \square

3 Arbitrary processing times, preemption and many calibration types

In this section, we consider a generalization of the model of Bender et al. in which there are more than one types of calibrations. Every calibration type is associated with a length l_i and a cost f_i . We are also given a set of jobs, each one characterized by its processing time p_j , its release time r_j and its deadline d_j . Our objective is to find a feasible preemptive schedule minimizing the total calibration cost. We prove that the problem is NP-hard.

Proposition 5. *The problem of minimizing the calibration cost is NP-hard for jobs with arbitrary processing times and many types of calibrations, even when the preemption is allowed.*

In order to prove the NP-hardness, we use a reduction from the well known UNBOUNDED SUBSET SUM problem (which is NP-hard [8], [9]). In an instance of the UNBOUNDED SUBSET

SUM problem, we are given a set of n items where each item j is associated to a value κ_j . We are also given a value V . We aim to find a subset of the items that sums to V under the assumption that an item may be used more than once.

Proof. Let Π be the preemptive scheduling problem of minimizing the total calibration cost for a set of n jobs that have arbitrary processing times in the presence of a set of K calibrations types.

Given an instance of the UNBOUNDED SUBSET SUM problem, we construct an instance of problem Π as follows. For each item j , create a calibration length $\ell_j = \kappa_j$ and of cost $f_j = \kappa_j$. Moreover, we create n jobs with positive arbitrary processing times such that $\sum_i p_i = V$ with $r_i = 0$ and $d_i = V \forall i$.

We claim that the instance of the UNBOUNDED SUBSET SUM problem is feasible if and only if there is a feasible schedule for problem Π of cost V .

Assume that the instance of the UNBOUNDED SUBSET SUM is feasible. Therefore, there exists a subset of items C' such that $\sum_{j \in C'} \kappa_j = V$. Note that the same item may appear several times. Then we can schedule all jobs, and calibrate the machine according to the items in C' in any arbitrary order. Since the calibrations allow all the jobs to be scheduled in $[0, V)$, then we get a feasible schedule of cost V for Π .

For the opposite direction of our claim, assume that there is a feasible schedule for problem Π of cost V . Let \mathcal{C} be the calibrations that have been used in the schedule. Then $\sum_{j \in \mathcal{C}} \ell_j = V$. Therefore, the items which correspond to the calibrations in \mathcal{C} form a feasible solution for the UNBOUNDED SUBSET problem. \square

4 Unit-time jobs, many calibration types and activation time

Given that the problem is NP-hard when many calibration types are considered even in the case where the calibrations are instantaneous, we focus in this section on the case where the jobs have unit processing times. We also assume that the activation time, that we denote by λ , is the same for every calibration type. As it has been pointed out in the introduction, for feasibility reasons, we allow to recalibrate the machine at any time point, even when it is already calibrated. It is easy to see that the introduction of the activation time in the model makes necessary the extension of the set of “important” dates that we have used in Section 2. Indeed, jobs can be scheduled at a distance bigger than n from a release date or a deadline. However, as we prove below, it is still possible to define a polynomial size time-set.

In the worst case, we have to calibrate n times and schedule n jobs. Thus the calibration can start at a time at most $n(\lambda + 1)$ time units before a deadline. Note that it is not necessary to consider every date in $[d_i - n(\lambda + 1), d_i]$ for a fixed i .

Definition 2. Let $\Theta := \bigcup_i \{d_i - j\lambda - h, j = 0, \dots, n, h = 0, \dots, n\}$.

Proposition 6. *There exists an optimal solution in which each calibration starts at a time in Θ .*

Proof. We show how to transform an optimal schedule into another schedule satisfying the statement of the proposition without increasing the total calibration cost. Let c_j be the last calibration that does not start at a date in Θ . We can shift this calibration to the right until:

- one job of this calibration finishes at its deadline and hence, it is no more possible to push this calibration to the right anymore. This means that there is no idle time between the starting time of this calibration and this deadline. Thus the starting time of this calibration is in Θ .
- the current calibration meets another calibration. In this case, we continue to shift the current calibration to the right while this is possible. Perhaps, there will be an overlap

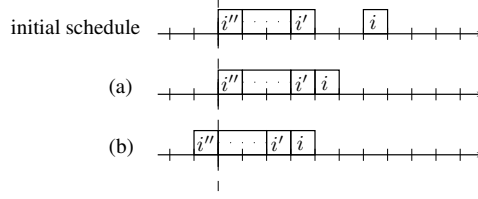


Figure 3: Illustration of Proposition 7

between calibration intervals, but as we said before, we allow to recalibrate the machine at any time. If we cannot shift to the right anymore, either a job ends at its deadline (and we are in the first case), or there is no idle time between the current calibration and the next one. Since there is at most n jobs and the next calibration starts at a time $d_i - j\lambda - h$ for some i, j, h , then the current calibration starts at a time $d_i - (j+1)\lambda - (h+h')$ where h' is the number of jobs scheduled in the current calibration with $h', h \leq n$.

□

Moreover the set of starting times of jobs has also to be extended by considering the activation time.

Definition 3. Let $\Phi := \{t + a \mid t \in \Theta, a = 0, \dots, n\} \cup_i \{r_i, r_i + 1, \dots, r_i + n\}$.

As for the starting time of calibrations, the worst case happens when we have to recalibrate after the execution of every job.

Proposition 7. *There exists an optimal solution in which the starting times and completion times of jobs belong to Φ .*

Proof. The first part of the proof comes from Proposition 1. Indeed, jobs can only be scheduled when the machine is calibrated. Let i be the first job that is not scheduled at a time in Φ in an optimal solution. Thanks to Proposition 1, we know that a calibration occurs before a deadline. Job i belongs to some calibration that starts at time $t \leq d_j$ for some j . By moving job i to the left, the cost of the schedule does not increase. Two cases may occur:

- job i meets another job i' (Fig. 3(a)). In this case, we consider the continuous block of jobs i'', \dots, i', i . We assume that at least one job in this block is scheduled at its release date and job i is at a distance at most n of this release date. Otherwise, we can shift this block of jobs to the left by one time unit (Fig. 3(b)). Indeed, this shifting is possible since job i'' is not executed at a starting time of a calibration (if it is the case, job i is in Φ by definition). Since job i' was in Φ , by moving this block, job i will be scheduled at a time in Φ .
- job i meets its release date.

□

Definition 4. Let $S(j, u, v) = \{i \mid i \leq j \text{ and } u \leq r_i < v\}$. We define $F(j, u, v, t, k)$ as the minimum cost of a schedule of the jobs in $S(j, u, v)$ such that:

- all these jobs are scheduled during the time-interval $[u, v)$
- the last calibration of the machine is at time t for a length of $\lambda + \ell_k$ (where the time-interval $[t, t + \lambda)$ corresponds to the activation time)
- the first calibration is not before u .

We are now ready to give our dynamic programming algorithm. We examine two cases depending on whether r_j belongs or not to the interval $[u, v]$.

Proposition 8. *One has $F(j, u, v, t, k) = F'$*

$$F' := \min \begin{cases} F(j-1, u, v, t, k) & \text{if } r_j \notin [u, v) \\ \min_{\substack{u' \in \Phi, r_j \leq u' < t' + \ell_{k'} + \lambda \\ t' \in \Theta, t' \leq u' \\ 1 \leq k' \leq K}} F(j-1, u, u', t', k') + F(j-1, u' + 1, v, t, k) & \text{otherwise} \end{cases}$$

with $F(0, u, v, t, k) := f_k, \forall t + \lambda \leq v \text{ \& } t \geq u$.

$F(0, u, v, t, k) := +\infty$ otherwise.

The optimal value is $\min_{t \in \Theta, 1 \leq k \leq K} F(n, \min_i r_i, \max_i d_i, t, k)$.

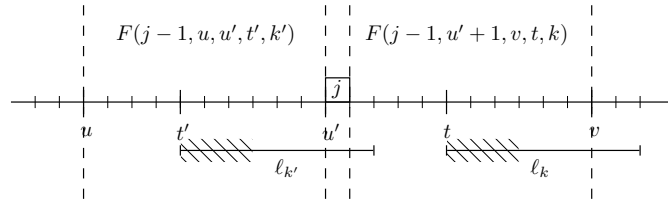


Figure 4: Illustration of Proposition 8

Proof. When $r_j \notin [u, v)$, we have necessarily $F(j, u, v, t, k) = F(j-1, u, v, t, k)$. In the following, we suppose that $r_j \in [u, v)$

We first prove that $F(j, u, v, t, k) \leq F'$.

We consider a schedule S_1 that realizes $F(j-1, u, u', t', k')$ and a schedule S_2 that realizes $F(j-1, u'+1, v, t, k)$. We build a schedule as follows: from time u to time u' use S_1 , then execute job j in $[u', u'+1)$, and finally from $u'+1$ to time v use S_2 . Moreover, it contains all jobs in $\{i \mid i \leq j \text{ and } u \leq r_i < v\}$. Since the first calibration in S_2 does not begin before $u'+1$, then we have a feasible schedule.

So $F(j, u, v, t, k) \leq F'$.

We now prove that $F(j, u, v, t, k) \geq F'$.

Since $j \in \{i \mid i \leq j \text{ and } u \leq r_i < v\}$, job j is scheduled in all schedules that realize $F(j, u, v, t, k)$.

Among such schedules, let X denote the schedule of $F(j, u, v, t, k)$ in which the starting time of job j is maximal. We claim that all jobs in $\{i \leq j, u \leq r_i < v\}$ that are released before or at u' are completed at u' . If it is not the case, we could swap the execution of such a job with job j , getting in this way a feasible schedule with the same cost as before. This will contradict the fact that the starting time of job j is maximal.

We consider a schedule S_1 that realizes $F(j-1, u, u', t', k')$ and a schedule S_2 that realizes $F(j-1, u'+1, v, t, k)$.

Then, the restriction of S_1 in X to $[u, u')$ will be a schedule that meets all constraints related to $F(j-1, u, u', t', k')$. Hence its cost is greater than $F(j-1, u, u', t', k')$. Similarly, the restriction of S_2 in X to $[u'+1, v)$ is a schedule that meets all constraints related to $F(j-1, u'+1, v, t, k)$.

Finally, $F(j, u, v, t, k) \geq F'$ \square

Proposition 9. *The problem of minimizing the total calibration cost with arbitrary calibration lengths, activation time and unit-time jobs can be solved in time $O(n^{16} K^2)$.*

Proof. This problem can be solved with the dynamic program in Proposition 8. Recall that the objective function is $\min_{t \in \Theta, 1 \leq k \leq K} F(n, \min_i r_i, \max_i d_i, t, k)$. The size of both sets Θ and Φ is $O(n^3)$. Indeed, by rewriting the set Φ , we have

$$\begin{aligned} \Phi &= \bigcup_i \{r_i, r_i + 1, \dots, r_i + n\} \cup \{t + a \mid t \in \Theta, a = 0, \dots, n\} \\ &= \bigcup_i \{r_i, r_i + 1, \dots, r_i + n\} \bigcup_i \{d_i - j\lambda - k + a, j = 0, \dots, n, k = 0, \dots, n, a = 0, \dots, n\} \\ &= \bigcup_i \{r_i, r_i + 1, \dots, r_i + n\} \bigcup_i \{d_i - j\lambda + k, j = 0, \dots, n, k = -n, \dots, n\} \end{aligned}$$

The size of the table is $O(n^{10}K)$. When each value of the table is fixed, the minimization is over the values u' , t' and k' , so the time complexity is $O(n^6K)$. Therefore the overall complexity time is $O(n^{16}K^2)$. \square

Note that when there is no feasible schedule, the objective function $\min_{t \in \Theta, 1 \leq k \leq K} F(n, \min_i r_i, \max_i d_i, t, k)$ will return $+\infty$.

5 Conclusion

We considered different extensions of the model introduced by Bender et al. in [4]. We proved that the problem of minimizing the total calibration-cost on a single machine can be solved in polynomial time for the case of jobs with arbitrary processing times when the preemption is allowed. Then we proved that the problem becomes NP-hard when there are many calibration types. Finally, we considered the case with many calibration types, where the calibrations are not instantaneous but take machine time, and we proved that the problem can be solved in polynomial time using dynamic programming for unit-time jobs. An interesting question is whether it is possible to find a lower time-complexity algorithm for solving this version of the problem, either optimally, or in approximation. Of course, it would be of great interest to study the case where more than one machines are available. Recall that the complexity of the simple variant studied by Bender et al. remains unknown for the multiple machines problem and that Fineman and Sheridan proposed approximation algorithms for the non-preemptive case when resource augmentation is allowed. It would be interesting to study the case where the preemption of the jobs is allowed.

References

- [1] New integrated stockpile evaluation program to better ensure weapons stockpile safety, security, reliability. <http://www.sandia.gov/LabNews/060331.html>, 2006.
- [2] Philippe Baptiste. Scheduling unit tasks to minimize the number of idle periods: a polynomial time algorithm for offline dynamic power management. In *Proceedings of the 17th Annual ACM-SIAM SODA 2006*, pages 364–367. ACM Press, 2006.
- [3] Philippe Baptiste, Marek Chrobak, and Christoph Dürr. Polynomial-time algorithms for minimum energy scheduling. *ACM Transactions on Algorithms*, 8(3):26, 2012.
- [4] Michael A. Bender, David P. Bunde, Vitus J. Leung, Samuel McCauley, and Cynthia A. Phillips. Efficient scheduling to minimize calibrations. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13*, pages 280–287. ACM, 2013.
- [5] Jeremy T. Fineman and Brendan Sheridan. Scheduling non-unit jobs to minimize calibrations. In Guy E. Blelloch and Kunal Agrawal, editors, *Proceedings of the 27th ACM on*

Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015, pages 161–170. ACM, 2015.

- [6] Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.
- [7] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. ACM*, 47(4):617–643, 2000.
- [8] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.
- [9] Oana Muntean and Mihai Oltean. Optical solutions for the unbounded subset-sum problem. *International Journal of Innovative Computing Information and Control*, 5(8):2159–2167, 2009.
- [10] Cynthia A. Phillips, Clifford Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.